BeyOND — Unleashing BOND

Thomas Bernecker, Franz Graf, Hans-Peter Kriegel, Christian Mönnig, Arthur Zimek
Institut für Informatik, Ludwig-Maximilians-Universität München
Oettingenstrasse 67
80538 München, Germany
{bernecker,graf,kriegel,zimek}@dbs.ifi.lmu.de
moennig@cip.ifi.lmu.de

ABSTRACT

While the abundance of data storage and retrieval systems is based upon horizontally decomposed data, occasionally studied vertical decompositions exhibit intriguing advantages but also bear serious disadvantages themselves. Here, we elaborate on the vertical decomposition technique employed in BOND [10], facilitating similarity search in high-dimensional data under certain restrictions. We report on our advances in overcoming some of the restrictions of BOND.

1. INTRODUCTION

It is common opinion, that similarity search in high-dimensional data is inherently difficult. The reasons for this finding, however, are not that widely agreed upon. For example, it has been stated that high-dimensional similarity search facilitated by partitioning or clustering based data structures cannot beat the sequential scan [22]. This has been backed but also relativized by some mainly theoretical studies [9, 4, 14, 2]. The essence of these studies for research on data structures is: it depends on the characteristics of the data distributions whether an index-based method is more suitable than a sequential scan-based method or vice versa. This may not seem impressively enlightening. Alas, surprisingly enough, this key message has been neglected in many research contributions over the last decade, examples being [1, 17, 5, 16, 11, 3]. Thus, it still appears to be well worth noting that nearest neighbor search is meaningful if and only if the nearest neighbor of the arbitrary query object is sufficiently different from its farthest neighbor. This is in general the case whenever a data set exhibits a natural structure in clusters or groupings of subsets of data, e.g., when the data are generated by several different distributions. It is, however, not well studied which impact the relation of relevant versus irrelevant attributes in a data space has. How do data structures behave, if the grouping of data is evident only in subspaces (built by the "relevant" attributes) of the original data space whereas "irrelevant" attributes do not contribute to discerning the different data groups from each other. Furthermore, if there exist several clusters within a data set, some attributes can be relevant for some clusters (i.e., useful for separation of these clusters) and at the same time irrelevant for other clusters. These important differentiations have been elaborated recently in [15, 8].

Established index structures [21] are designed and optimized for the complete data space where all attributes contribute to partitioning, clustering etc. For these data structures, the space of queries facilitated by the index structure must be fixed prior to the construction of the index structure. Approaches addressing the problem of subspace similarity search *explicitly* are [20, 18]. There, the authors propose an adaptation of the VA-file [22] to the problem of subspace similarity search. The basic idea of these approaches is to split the original VA-file into d *partial* VA-files, where d is the data dimensionality, i.e. we get one file for each dimension containing the approximation of the original full-dimensional VA-file in that dimension. Based on the information of the partial VA-files, upper and lower bounds of the true distance between data objects and the query are derived. Subspace similarity queries are processed by scanning only the relevant files in the order of relevance, i.e. the files are ranked by the selectivity of the query in the corresponding dimension. As long as there are still candidates that cannot be pruned or reported using the upper and lower distance bounds, the next ranked file is read to improve the distance approximations or (if all partial VA-files have been scanned) the exact information of the candidates accessed to refine the exact distance. Another approach to the problem is proposed in [19], although only ε-similarity range queries are supported. The idea of this method is based on multiple pivot-points to derive lower and upper bounds for distances. The bounds are computed in a preprocessing step for a couple of pivot points. Essentially, this approach allows to sequentially scan the database reading only the information on lower and upper bounds and to refine the retrieved candidates in a postprocessing step. A bottom-up combination of one-dimensional indices and a top-down search in a full-dimensional index structure, restricted according to the query, are discussed in [7, 6].

As opposed to all these approaches, BOND [10] is essentially also a search strategy for the full-dimensional space enhancing the sequential scan. It is, however, quite naturally possible to restrict a query to a given subspace, since the basic idea of BOND is to use a column store (as it might be known from NoSQL database systems). BOND ranks the columns according to their potential impact on distances and prunes later columns if their impact becomes too small to change the query result. By the design of this method, subspace queries can be *implicitly* facilitated with the same architecture. However, BOND is motivated by the application of metrics for image retrieval and, thus, requires certain properties of a data set which restricts the application considerably:

1. The first metric proposed in BOND is only applicable to normalized histogram data.

2. Using Euclidean distance, still the length of each vector is required for pruning columns with low impact.

3. Stricter bounds for the Euclidean distance metric further improve the pruning, but require Zipfian distributed data (like color or gray scale histograms) and a certain resolve order of the columns in the database.

In this paper, we are specifically interested in extending BOND by loosening the restrictions of its use for data sets and by improving the pruning power. In the following, we will detail BOND and its deficiencies w.r.t. these aspects (Section 2), before we describe our extensions (Section 3). We will then demonstrate the improved performance (Section 4) and conclude in Section 5.

2. BOND REVISITED

Processing multi-step queries using a filter-refinement framework, traditional index approaches resolve the data of feature vectors row-wise (horizontally) in order to obtain their exact representation. The main advantage of BOND is that feature vectors are resolved column-wise (vertically) so that the values of a feature vector v are obtained successively. Thus, the resolved part of the feature vector is known exactly whereas the unresolved part has to be approximated. This approach is inherently different from traditional tree-indexing approaches where a feature vector is either completely approximated or completely available. In order to avoid possibly unnecessary IO-operations, traditional tree-indexing techniques aim at avoiding to resolve as many feature vectors as possible which are not part of the result set. On the contrary, BOND starts with resolving all feature vectors column by column and tries to approximate the remaining part of the feature vector. As soon as the approximation yields a sufficiently high pruning power, false candidate feature vectors can be pruned from the candidate set, so that the remaining dimensions of these feature vectors need not be resolved. BOND supports regular k-NN queries on the full data set as well as on weighted subspaces. Nevertheless, the pruning bounds deteriorate in case of subspace queries.

The main goal of the pruning statistics used in BOND is to tighten the approximations of the yet unresolved parts of the feature vector in order to be able to prune false candidates from the candidate set as soon as possible before resolving additional columns for this vector.

In the rest of the paper, we follow the notation of [10], where $q \in \mathbb{R}^d$ denotes a d-dimensional query vector and $v \in \mathbb{R}^d$ denotes an arbitrary d-dimensional feature vector of the database. Furthermore, any database vector v can be split into a resolved part $v^- \in \mathbb{R}^m$ and an unresolved part $v^+ \in \mathbb{R}^{d-m}$, so that $v = v^- \cup v^+$. The variable $m \in [1, d]$ denotes the amount of columns that have been resolved so far. The distance $S(q, v)$ between q and v can thus be approximated by a composition of the exact distance plus the approximation:

$$S_{approx}(q, v) = S_1(q^-, v^-) + S_2(q^+, v^+) \qquad (1)$$

Assuming a k-nearest neighbor (k-NN) query, the resulting distance bounds are then used to refine the candidate set in a traditional way, where all candidates are pruned if their lower distance bound is greater than the kth smallest upper bound. The distance $S(q^-, v^-)$ between the known parts of q and v can be computed precisely. Concerning the unknown part (v^+), an approximation for the lower and upper distance bounds to the query vector q needs to be created. The computation of $S(q^+, v^+)$ of course depends on whether the upper or lower bound has to be computed.

The basic approach of BOND uses the application scenario of histogram data, where the length of each data vector can safely be assumed to be 1. Relaxing this condition, an extension of the basic approach assumes the unit hypercube $[0, 1]^d$ as the data space. This extension is based on the Euclidean distance between the query vector q and the database vector v and does not rely on any distribution or assumption of the data set, as it only depends on q. Thus, the exact distance S_1 and the upper approximation S_2 between q and v are derived as follows:

$$S_1(q, v) = \sum\nolimits_i (q_i - v_i)^2 \qquad (2)$$

$$S_2(q, v) = \sum\nolimits_i \max\{q_i, 1 - q_i\}^2 \geq S_1(q, v) \qquad (3)$$

Using the approximated distance of Eq. 1, the obtained bounds are:

$$S_{upper}(q, v) = S_1(q^-, v^-) + S_2(q^+, v^+) \geq S_1(q, v) \qquad (4)$$

$$S_{lower}(q, v) = S_1(q^-, v^-) + 0 \leq S_1(q, v) \qquad (5)$$

The advantage of the independence from the data distribution and resolve order is paid with the loss of pruning power. The weakness of these bounds is obvious, especially for the lower bound which assumes the distance 0 for the remaining, unresolved subspace and the upper bound only takes the query vector into account and does not make any assumptions on the database vector. A second extension relies on the precomputed length of each feature vector v, which is stored in the database additionally to the values of v, and a skewed Zipfian distribution of the data set. This method is used as a reference as it provided the best results in the original paper. In this case, a large number of distance computations and IO-operations can be saved compared to the sequential scan. However, the upper and lower distance bounds computed by this method quickly lose their pruning power if the data distribution changes. Also this method strictly requires a certain resolve order of the columns in the database, which is not optimal in the case of other distributions or in case of correlated dimensions. Changing the resolve order however is not an option, because this would invalidate the proof for the correctness of the pruning bounds.

3. BEYOND BOND

One of the main limitations of BOND is the dependence on the data distribution. The distance approximations proposed in [10] work well as long as the data follows a skewed Zipfian distribution like in the case of color histograms and if the database columns are resolved in decreasing order of the query feature values. If either of the conditions is not fulfilled, BOND quickly degenerates, i.e. the complete data set needs to be resolved to answer the query. Thus, BeyOND extends the original idea of BOND in order to supply a query system that allows an efficient execution of k-NN queries on data sets that follow an arbitrary or unknown distribution, so that the following restrictions are removed:

1. BeyOND does not depend on the data distribution, so any distance metric can be employed that provides valid upper and lower distance approximations.
2. The values v_i of the feature vectors are no more restricted to $v_i \in [0, 1]$ in each dimension.
3. BeyOND does not rely on a specific resolve order of the query vector, so more sophisticated resolve techniques can be applied to further increase the pruning power.

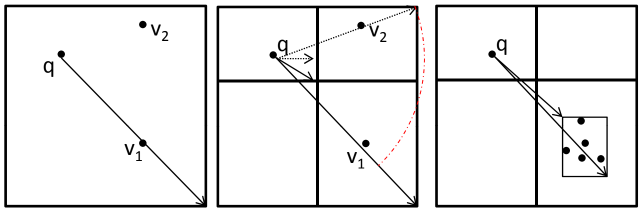
Removing the first and third restriction also disables the possibility to use the improved distance approximations of the original work. Thus, we have to start by improving the weak approximations shown in Equations 4 and 5.

In the following, we describe how BeyOND combines the concepts of BOND and the VA-file [22] by introducing *Sub Cubes* (Section 3.1), supported by minimum bounding rectangles (MBRs) for certain sub cubes (Section 3.2), based on a BOND-style column-store architecture. This way, it is still possible to resolve the data set in a column wise manner. A restriction that remains in BeyOND, however, is the embedding into a non-unit equilateral hyper cube, so that the minimum and maximum values of each dimension need to be known.

3.1 Sub Cubes

The first extension we propose is to pick up the idea of the VA-file [22] by splitting the cube once in each dimension. We thus partition the hyper cube describing the feature space into 2^d pairwise disjunct sub cubes. Each sub cube can be identified by the according Z-Order ID (Z-ID), which is stored as a memory-efficient bit-representation. This Z-ID is stored additionally to the values of each feature vector. The locations of the split positions in each dimension are stored in separate arrays, so that quantile splits are also supported. Assuming that the feature vectors are composed of 8 byte double values, the memory consumption of a feature vector increases by a value of $\lceil \frac{sd}{8} \rceil$ bytes with s denoting the amount of split levels. It would also be possible to increase the split level of the cubes even further. Nevertheless, each additional split also directly increases the size of the Z-IDs. This leads to a trade-off between additional memory consumption from larger Z-IDs and tighter approximations for the upper and lower bounds of the distance computation due to smaller sub cubes. An evaluation about the impact of additional split levels is shown in the evaluation (Sec. 4). Given a Z-ID of a feature vector and the coordinate arrays containing the split positions, it is a computationally cheap task to recreate the coordinates of the according sub cube, so that the MBRs of potentially 2^d sub cubes need not be kept in memory but can be quickly recomputed on demand.

The sub cubes provide the advantage that the upper and lower distance approximations need not be computed w.r.t. the complete hyper cube that encloses the feature space but only between the cubes containing the query vector and the feature vectors of the database. Thereby, we need to consider the following two cases: Let Z_q and Z_v be the Z-IDs of the query vector q and a vector v of the database.



(a) No split. (b) One split. (c) Split with MBR.

Figure 1: Improvement of the upper/lower distance approximation.

$Z_q = Z_v$ indicates that both q and v share the same sub cube, so the upper bound of the distance approximation can be lowered to the borders of this sub cube (Eq. (9)). The lower distance remains 0 for all unresolved dimensions.

$Z_q \neq Z_v$ implies that q and v are located in different cubes, so the lower distance approximation can be raised to the minimum distance of q to the sub cube containing v (Eq. (8)). The upper distance approximation is again computed w.r.t. the bounds of the hyper cube containing v using Equation (9). Compared to approximating the upper distance w.r.t. the complete hyper cube, this decreases the upper bound when both sub cubes share a common plane, which is the case in $d-2$ out of $d-1$ cases (cf. Fig. 1(a) and 1(b)).

$$S_2^{'}(q,v) = \sum \max \left\{ |q_i - c_{v_i}^{lower}|, |q_i - c_{v_i}^{upper}| \right\}^2 \tag{6}$$

$$S_2^{''}(q,v) = \sum \begin{cases} 0, & \text{if } q_i \in \left[c_{v_i}^{lower}, c_{v_i}^{upper}\right] \\ \min \left\{ |q_i - c_{v_i}^{lower}|, |q_i - c_{v_i}^{upper}| \right\}^2 \end{cases} \tag{7}$$

$$S_{lower}^{'}(q,v) = S_1(q^-, v^-) + S_2^{''}(q^+, v^+) \geq S_1(q,v) \tag{8}$$

$$S_{upper}^{'}(q,v) = S_1(q^-, v^-) + S_2^{'}(q^+, v^+) \leq S_1(q,v) \tag{9}$$

where $c_{b_i}^{upper}$ ($c_{b_i}^{lower}$) denotes the upper (lower) bound of the sub cube c containing the feature vector v in dimension i.

3.2 MBR Caching

In high-dimensional data sets that do not cluster strongly, the majority of the 2^d sub cubes is occupied by at most one feature vector. In the few cases that a sub cube is occupied by more feature vectors, we propose to tighten the approximation of the sub cubes using MBRs (cf. Fig. 1(c)). Therefore, we iterate through all sub cubes which are occupied by more than one feature vector, compute the MBR for the according set of feature vectors and store this MBR in a priority queue (PQ) which is sorted in descending order w.r.t. the ranking function

$$f(\text{MBR}) = \frac{V_{\text{sub cube}} \cdot card(\text{MBR})}{V_{\text{MBR}}} \tag{10}$$

where $card(\text{MBR})$ denotes the number of feature vectors contained in the according MBR and V denotes the volume of the sub cube or MBR.

As the resulting MBRs cannot be derived from any fixed values similar to the case of the split positions, at least 2

Table 1: Data sets used in the evaluation.

Name	Cols.	Rows	Distribution
ALOI	27	110 250	Zipfian
CLUSTERED	20	500 000	50 clusters
PHOG	110	10 715	gradient histograms

d-dimensional coordinates are required to define each MBR, so that each MBR requires $d \cdot 16$ bytes (again assuming 8 byte double coordinates). Even though this seems to be a quite large overhead, an MBR can be shared among all feature vectors of the respective set. Thus, the memory increase is reduced to $\frac{d \cdot 16}{card(\text{MBR})}$ per feature vector comprised by the MBR. As the MBR is associated with the respective Z-ID, not even an additional memory pointer is required for the feature vector.

In order to define an upper limit for this additional memory consumption, we limit the size of the MBR queue PQ to 1% of the amount of total feature vectors in the database. Combined with the ranking function Eq. (10), we ensure that we hold only a limited amount of MBRs which contain a large amount of feature vectors on the one hand and also a significantly smaller volume compared to the surrounding sub cube on the other hand. This threshold has to be chosen as a trade-off between pruning power and additional memory consumption. Alternatively, the threshold can also be chosen in absolute values if the maximum amount of memory should be limited. In any case, the threshold should be chosen low enough so that either all MBRs can be kept in memory or it should be ensured that only those MBRs are read from disk that approximate a fairly large amount of feature vectors, so that the time needed to load the MBRs is still smaller than resolving the respective feature vectors.

In order to use the tighter approximation provided by the MBRs, the variables $c_{b_i}^{lower}$ and $c_{b_i}^{upper}$ in Eqs. (6) and (7) need to be filled with the coordinates of the MBR instead with those of the sub cube, so that this second extension integrates seamlessly into the computation of the distance approximations.

4. EXPERIMENTS

4.1 Data Sets

In our experiments, we evaluated the proposed techniques on three data sets (summarized in Table 1):

First, we used 27-dimensional color histograms extracted from the ALOI data set [13] comprising 110,250 feature vectors (ALOI). This data set poses the hardest challenge as BOND is expected to perform best on this data set as the color histograms follow a Zipfian distribution. Second, we used a 20-dimensional synthetic clustered data set, comprising 500,000 feature vectors organized in 50 clusters, each following a 20-dimensional Gaussian distribution (CLUSTERED). Finally, we used a data set from the area of medical imaging, containing 10,715 feature vectors with 110 dimensions (PHOG). The features were provided from the work of [12] and represent gradient histograms which were extracted from medical computer tomography images. The features were already reduced in dimensionality by applying a principal component analysis and the dimensions are ordered by decreasing value of the eigenvalues.

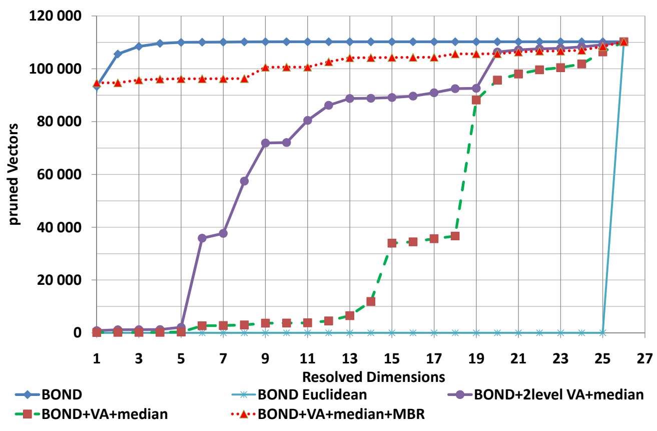


Figure 2: Pruning power on ALOI.

4.2 Evaluation

In order to measure the impact of the VA-file approach and the MBR caching, we evaluated the following tests: We evaluated both distance approximations of the original implementation of BOND using the improved distance approximation (BOND) and the simple approximation (BOND Euclidean). Then we evaluated the contribution of the VA-File-approach by measuring the pruning power of a 1- and a 2-level VA-file (BOND+VA+median, BOND+2level VA+median). Finally, we tested the additional impact by adding the MBR caching (BOND+VA+median+MBR).

In our experiments we submitted 50 k-NN ($k = 10$) queries to the database and measured the amount of feature vectors that were pruned after a data column was resolved and the distance approximations were recomputed. The measurements shown in the Figures 2, 3 and 4 represent the averaged cumulative amount of feature vectors that were pruned after a column was resolved. The area under the curves can thus be regarded as the amount of data that does not need to be resolved from disk, whereas the area above the curve indicates the amount of data that needs to be taken into account for further refinement from disk and for computation of the distance approximations. Thus, better approximations of the upper and lower distance bounds yield better pruning power, so that more feature vectors can be pruned at an early stage of the computation.

In the ideal case, only a few columns have to be resolved until the k final nearest neighbors remain in the data set. Also, the final aim of the algorithm is to prune as many feature vectors as possible at a very early stage of the algorithm so that further data columns of this feature vector do not have to be resolved.

Comparing the ALOI data set with the other data sets, it can be seen that the original BOND performs as expected on histogram-like data sets. Nevertheless, BOND resolves about half of the data on the CLUSTERED data set and almost all columns on the PHOG data set. This shows the strong dependence on the data distribution, which is addressed in this paper.

In the first step, we proposed to refine the simple Euclidean distance approximation by using the sub cubes that were derived from the Z-ID which was saved additionally to the feature vector. While the improvement in the ALOI data set is clearly visible, the impact on the CLUSTERED

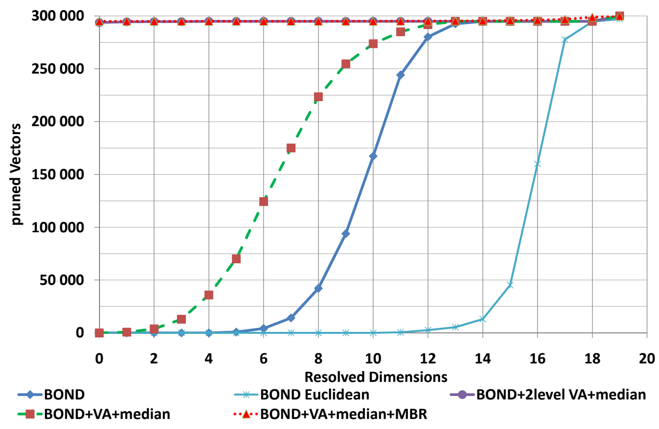


Figure 3: Pruning power on CLUSTERED.

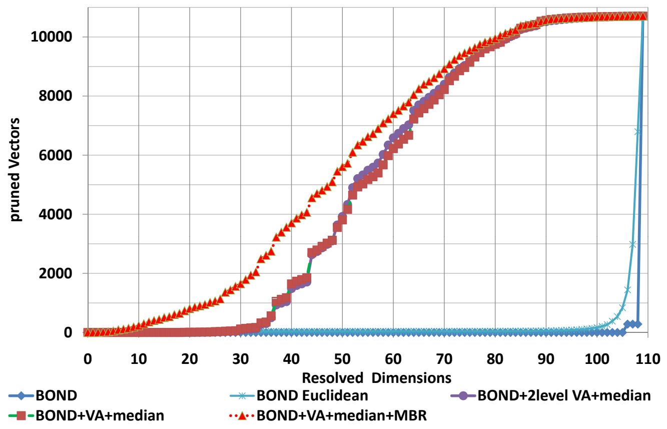


Figure 4: Pruning power on PHOG.

Table 2: Table showing the pruning power of the sub cubes. The columns show the data set, the amount of splits per dimension and the amount of resolved columns (in percent), where more than 25%, 50% and more than 90% of the candidates were pruned.

Data set	Splits	25%	50%	90%
ALOI	1	16 (59%)	19 (70%)	23 (85%)
CLUSTER	1	7 (35%)	8 (40%)	10 (50%)
PHOG	1	45 (41%)	58 (53%)	80 (73%)
ALOI	2	7 (26%)	9 (33%)	21 (75%)
CLUSTER	2	1 (5%)	1 (5%)	1 (5%)
PHOG	2	45 (41%)	55 (50%)	79 (72%)

and PHOG data sets is much higher (cf. Table 2, rows 1-3). Here, the amount of resolved dimensions is lower using the CLUSTERED and PHOG data sets compared to the ALOI data set.

The intuitive approach to add more splits per dimension and thus decrease the size of the sub cubes performs well with ALOI and CLUSTERED. Nevertheless, the improvement with PHOG (cf. Table 2, rows 3-6) is negligible, while obviously CLUSTERED takes most advantage from the quadratic growth of additional sub cubes ($2^d \rightarrow 4^d$), which poses a very good approximation of the clusters.

Table 3: Table showing the pruning power of Sub Cubes + MBRs. The columns show the data set, and the amount of resolved columns (in percent), where more than 25%, 50% and more than 90% of the candidates were pruned.

Data set	25% pruned	50% pruned	90% pruned
ALOI	1 (4%)	1 (4%)	10 (37%)
CLUSTER	1 (5%)	1 (5%)	1 (5%)
PHOG	37 (34%)	50 (45%)	77 (70%)

Table 4: Total amount of data viewed. The columns show the data set and the amount of data resolved in case of 1 and 2 splits per dimensions and the combination of 1 split and cached MBRs.

Data set	1 split	2 splits	1 split + MBR
ALOI	66.9%	38.3%	7.7%
CLUSTER	34.1%	1.6%	1.4%
PHOG	52.6%	52.3%	45.4%

The second improvement pre-computes the MBRs in the case a sub cube contains more than a single feature vector, the MBR would be small enough and the maximum amount of MBRs is not reached yet. More sophisticated methods to determine the maximum amount of MBRs could regard the vector distribution within the cube, a minimum volume decrease, etc. In this case, we used 1% of the amount of feature vectors as a limit for the number of created MBRs for the sub cubes with the largest volume decrease. Also, each dimension was just split once. The result can be seen in the Figures 2, 3 and 4, indicated by the dotted line, and in Table 3. Using the ALOI data set, the initial pruning power in dimension 1 is even comparable to the original BOND method. The CLUSTERED data set performs best as before, as 98% of the data could be pruned at once. PHOG again poses the hardest challenge. Yet, there is still an improvement compared to the basic sub cube approach (with 1 or 2 splits per dimension).

Table 4 shows the total amount of data that was resolved from the data set. It can be seen that in case of ALOI and PHOG, it is more profitable to extend the original idea of BOND with a 1-level VA-file (1 split per dimension) using the technique of MBR caching instead of simply adding more layers (2 or more splits per dimension) which generates more sub cubes. Using the CLUSTERED data set, there is almost no difference between the approaches of more splits and MBR caching. Nevertheless, the solution of a single split combined with MBRs offers more flexibility regarding the choice of MBRs and the control of additional memory consumption than simply adding more splits.

5. CONCLUSION

In this paper, we presented the current state of our work of extending a technique for vertically decomposed data, known as BOND [10]. The proposed extensions supply the vertical decomposition and fast indexing of high-dimensional feature vectors now even if the data does not follow a certain distribution. We evaluated our extensions on real-world data and showed the superior performance compared to the prior work. For future work, we plan to further evaluate the trade-off between split level and pruning power as well as a modified resolve order depending on the query vector. We plan to develop more sophisticated techniques for the creation of the MBRs and the limitation of the amount of pre-computed MBRs. We also aim at finding solutions to abandon the restriction that the minimum and maximum values of the feature vectors need to be known in advance.

Acknowledgements
This research has been supported in part by the THESEUS program in the MEDICO and CTC projects. They are funded by the German Federal Ministry of Economics and Technology under the grant number 01MQ07020. The responsibility for this publication lies with the authors.

6. REFERENCES

[1] C. C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *ACM SIGMOD Record*, 30(1):13–18, 2001.

[2] C. C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory (ICDT), London, UK*, 2001.

[3] C. C. Aggarwal and P. S. Yu. On high dimensional indexing of uncertain data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico*, 2008.

[4] K. P. Bennett, U. Fayyad, and D. Geiger. Density-based indexing for approximate nearest-neighbor queries. In *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA*, 1999.

[5] S. Berchtold, C. Böhm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. Independent Quantization: An index compression technique for high-dimensional data spaces. In *Proceedings of the 16th International Conference on Data Engineering (ICDE), San Diego, CA*, 2000.

[6] T. Bernecker, T. Emrich, F. Graf, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. Subspace similarity search: Efficient k-nn queries in arbitrary subspaces. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany*, 2010.

[7] T. Bernecker, T. Emrich, F. Graf, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. Subspace similarity search using the ideas of ranking and top-k retrieval. In *Proceedings of the 26th International Conference on Data Engineering (ICDE) Workshop on Ranking in Databases (DBRank), Long Beach, CA*, 2010.

[8] T. Bernecker, M. E. Houle, H.-P. Kriegel, P. Kröger, M. Renz, E. Schubert, and A. Zimek. Quality of similarity rankings in time series. In *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN*, 2011.

[9] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT), Jerusalem, Israel*, 1999.

[10] A. P. de Vries, N. Mamoulis, N. Nes, and M. Kersten. Efficient k-NN search on vertically decomposed data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI*, 2002.

[11] J. Dittrich, L. Blunschi, and M. A. V. Salles. Dwarfs in the rearview mirror: How big are they really? In *Proceedings of the 34nd International Conference on Very Large Data Bases (VLDB), Auckland, New Zealand*, 2008.

[12] T. Emrich, F. Graf, H.-P. Kriegel, M. Schubert, M. Thoma, and A. Cavallaro. CT slice localization via instance-based regression. In *Proceedings of the SPIE Medical Imaging 2010: Image Processing (SPIE), San Diego, CA*, volume 7623, page 762320, 2010.

[13] J. M. Geusebroek, G. J. Burghouts, and A. Smeulders. The Amsterdam Library of Object Images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[14] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt*, 2000.

[15] M. E. Houle, H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany*, 2010.

[16] H. Jin, B. C. Ooi, H. T. Shen, C. Yu, and A. Y. Zhou. An adaptive and efficient dimensionality reduction algorithm for high-dimensional indexing. In *Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India*, 2003.

[17] N. Katayama and S. Satoh. Distinctiveness-sensitive nearest-neighbor search for efficient similarity retrieval of multimedia information. In *Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany*, 2001.

[18] H.-P. Kriegel, P. Kröger, M. Schubert, and Z. Zhu. Efficient query processing in arbitrary subspaces using vector approximations. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM), Vienna, Austria*, 2006.

[19] X. Lian and L. Chen. Similarity search in arbitrary subspaces under Lp-norm. In *Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico*, 2008.

[20] W. Müller and A. Henrich. Faster exact histogram intersection on large data collections using inverted VA-files. In *Proceedings of the 3rd International Conference on Image and Video Retrieval (CIVR), Dublin, Ireland*, 2004.

[21] H. Samet. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann, San Francisco, 2006.

[22] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York City, NY*, 1998.